Московский Государственный Университет им. М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Суперкомпьютеров и Квантовой Информатики

Отчёт

Параллельная реализация решения СЛАУ с помощью метода отражений.

Работу выполнил:
Лесцов Б.А.
423 группа

2018

Постановка задачи и формат данных.

Задача: Реализовать параллельный алгоритм решения СЛАУ с помощью метода отражений.

Постановка задачи:

Заданы невырожденная вещественная матрица *A* размера *NxN* и вещественный вектор *b* длины N такие что система *Ax=b* имеет единственное решение. Необходимо найти вектор *x,* являющийся решением этой системы.

Компиляция:

1) С помощью gnu make:

    > make

2) С помощью cmake:

    > mkdir build
    > cd build
    > cmake ..

Запуск:

1) Считать матрицу из файла:

    > mpirun -np 'число процессов' build/bin/Prak 'путь к файлу с матрицей'

Пример*:*

    > mpirun -np 2 build/bin/Prak data/mat.txt

2) Сгенерировать с помощью функции f(i, j), заданной в source/main.cpp:

    > mpirun -np 'число процессов' build/bin/Prak 'N' 'N+1', где N – размер матрицы системы

Пример*:*

    > mpirun -np 4 build/bin/Prak 1024 1025

Формат входных файлов:

Входной файл:

1) Файл с матрицей размера *Nx(N+1)* в текстовом виде. В начале файла располагаются два числа m, n типа size_t – размеры матрицы. Далее следуют n*m вещественных чисел – сама матрица. Последний столбец этой матрицы – это вектор *b.*

Формат входных файлов:
В стандартный поток вывода будет выдано решение заданной СЛАУ в формате:

x_1 = ‘значение переменной x_1’
…
x_i = ‘значение переменной x_i’
…
x_N = ‘значение переменной x_N’

Далее следует строка:

Mat_size ‘размер матрицы’ Comm_size ‘число процессов’
Forward_Time_(microsec) ‘время приведения к верхнетреугольному виду в микросекундах’
Backward_Time_(microsec) ‘время обратного хода метода Гаусса’ diff ‘невязка’

Описание алгоритма:
Метод отражений основан на разложении матрицы A системы *Ax=b* в произведение унитарной матрицы на верхнюю треугольную. Матрица A называется унитарной, если она удовлетворяет уравнению $A \cdot A^* = E$, где A^* - матрица, сопряженная с A. Вещественные унитарные матрицы называются ортогональными.

По своей структуре метод отражений близок к методу Гаусса, но исключение проводится с помощью матриц отражения, которые являются унитарными и эрмитовыми. Достоинством метода отражений является единая схема вычислительного процесса, не зависящая от структуры матрицы.

Теорема. Пусть s и l произвольные вектор-столбцы, причем вектор l имеет единичную длину. Тогда найдется такой вектор w, что построенная по нему матрица отражения $U = E - 2ww^*$ переведет вектор s в вектор, коллинеарный вектору l, т.е. $Us = \alpha l$.

Вектор w строится по правилу $w = \frac{1}{\rho}(s - \alpha l)$, где $|\alpha| = \sqrt{(\alpha, \alpha)}$, $\arg\alpha = \arg(s,l) - \pi$, $\rho = \sqrt{(s - \alpha l, s - \alpha l)} = \sqrt{2|\alpha|^2 + 2|\alpha||(s,l)|}$.

Будем преобразовывать расширенную матрицу систему по правилу

$A_{k+1} = U_{k+1} A_k$, $k = 0, 1, \ldots, n-2$

с помощью умножения слева на последовательность матриц отражения $U_1, U_2, \ldots, U_{n-1}$. Для построения матрицы U_1 на первом шаге метода в качестве вектора S берется первый столбец расширенной матрицы, а в качестве вектора l - координатный вектор $l = (1, 0, 0, \ldots, 0)^{\prime}$. В силу выбора векторов S и l все координаты первого столбца расширенной матрицы, кроме первой, после выполнения первого шага метода будут равны нулю.

Пусть уже построена матрица A_k, у которой $a_{i,j}^{(k)} = 0$, $i > j$, $j = \overline{1,k}$. Теперь в качестве S и l берутся вектора

$$s = (0, \ldots, 0, a_{k+1,k+1}^{(k)}, a_{k+2,k+1}^{(k)}, \ldots, a_{n,k+1}^{(k)})^{\prime}, \quad l = (0, \ldots, 0, 1, 0, \ldots, 0)^{\prime},$$

где в векторе l единица стоит на $k+1$-ом месте. После выполнения k-го шага метода отражений получим матрицу A_{k+1}, у которой все элементы, стоящие ниже главной диагонали, в первых $k+1$-ом столбцах будут равны нулю. Невозможность выполнения очередного шага связана только с равенством нулю вектора S, а это невозможно, так как матрица A является невырожденной.

После $(n-1)$-шага получим матрицу, первые n столбцов которой образуют верхнюю треугольную матрицу L. Система уравнений, соответствующая полученной расширенной матрице, равносильна исходной системе. Значения неизвестных находятся аналогично обратному ходу метода Гаусса

$$x_n = -\frac{a_{n,n+1}^{(n-1)}}{a_{n,n}^{(n-1)}}, \quad x_i = -\frac{a_{i,n+1}^{(n-1)} + \sum_{j=i+1}^{n} a_{i,j}^{(n-1)} x_j}{a_{n,n}(n-1)}, i = n-1, n-2, \ldots, 1$$

Параллельная версия алгоритма подразумевает разделение матрицы А между процессами по столбцам. При этом на каждом этапе работы алгоритма один из процессов подсчитывает вектор *w,* и рассылает его остальным процессам. Получив нужный вектор, каждый процесс производит обновление всех столбцов своей части матрицы по правилу: y_i *-= 2*alpha*w, где alpha -* скалярное произведение w и y_i .

Во время параллельного выполнения обратного хода метода Гаусса процессы последовательно вычисляют переменные x_i, после чего отсылают остальным процессам вычеселнный x_y, а также вектор x_i*y_i

где y_i – i-й столбец верхнетреугольной матрицы.

Результаты выполнения.

Тестирование производилось на системе Blue Gene/P. Использовались матрицы размеров 1024 x 1024, 2048 x 2048, 4096 x 4096 и 8192x8192. Для 8192x8192 приведены результаты для 128, 256, 512 и 1024 процессов, так как на меньшем количестве процессов программа работает слишком долго.

Результаты:

Число процессов	Прямой ход	Обратный ход	Общее время	Невязка
Размер матрицы: 1024x1024				
1	6.722922	0.018040	6.740962	0.000051
2	3.299157	0.014167	3.313324	0.000010
4	1.722122	0.014498	1.736620	0.000107
8	0.878477	0.011157	0.889634	0.000012
16	0.474213	0.009077	0.483290	0.000058
32	0.264235	0.008723	0.272958	0.000030
64	0.161500	0.008833	0.170333	0.000034
128	0.108091	0.009212	0.117303	0.000117
256	0.080308	0.009623	0.089931	0.000012
512	0.066610	0.012077	0.078687	0.000011
1024	0.058774	0.014110	0.072884	0.000062
Размер матрицы: 2048x2048				
1	58.058826	0.185454	58.244280	0.000073
2	29.406341	0.158643	29.564984	0.001528
4	14.656131	0.158322	14.814453	0.000219
8	7.610560	0.071177	7.681737	0.000398
16	4.059188	0.022106	4.081294	0.000068
32	2.030635	0.018013	2.048648	0.000072
64	1.113021	0.017605	1.130626	0.000066
128	0.665552	0.017770	0.683322	0.000059
256	0.438169	0.017623	0.455792	0.000094
512	0.323093	0.019399	0.342492	0.000051
1024	0.263822	0.021516	0.285338	0.000452

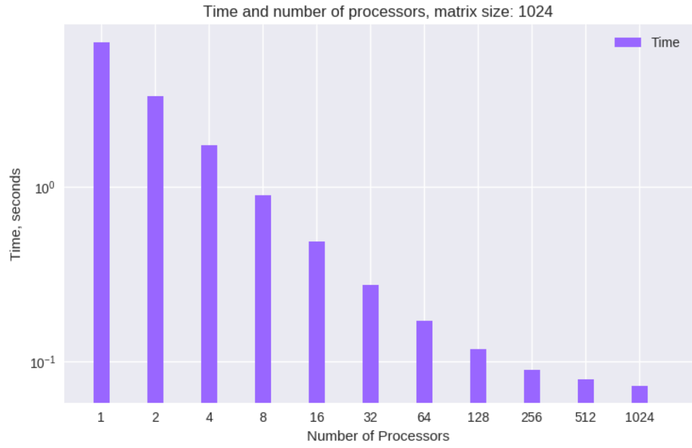
Число процессов	Прямой ход	Обратный ход	Общее время	Невязка
Размер матрицы: 4096x4096				
1	530.84392	0.749686	531.59361	0.000107
2	265.82526	0.635387	266.46064	0.000056
4	160.59724	0.637392	161.23463	0.000056
8	80.594747	0.331587	80.926334	0.000609
16	40.772724	0.172879	40.945603	0.000153
32	20.792026	0.082310	20.874336	0.038190
64	10.655758	0.037592	10.693350	0.000635
128	5.685375	0.035878	5.721253	0.000051
256	3.439362	0.035092	3.474454	0.000035
512	2.122923	0.036788	2.159711	0.000094
1024	1.488376	0.038696	1.527072	0.000218
Размер матрицы: 8192x8192				
128	42.617031	0.112642	42.729673	0.004127
256	23.868935	0.079526	23.948461	0.001467
512	13.053445	0.077279	13.130724	0.000168
1024	8.595275	0.076496	8.671771	0.000999

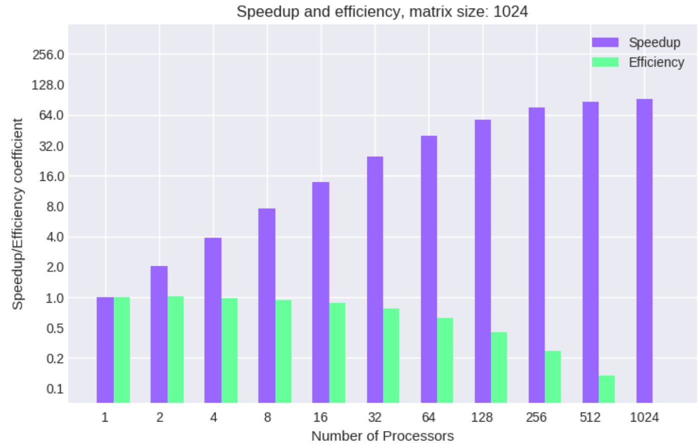
Ускорение и эффективность:

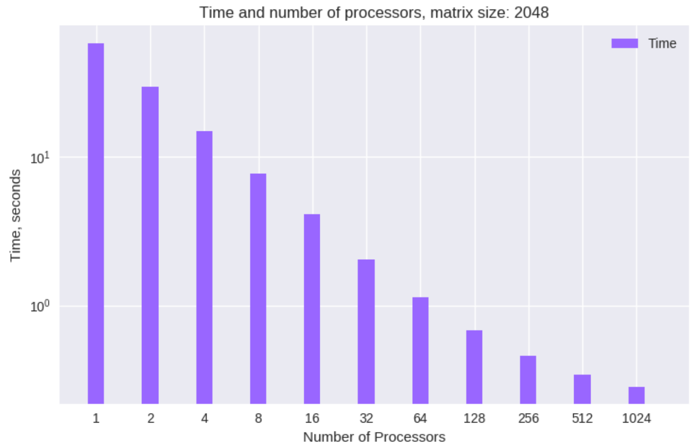
Число процессов	Ускорение	Эффективность	Ускорение	Эффективность
	1024x1024		**2048x2048**	
1	1.0	1.000000	1.0	1.000000
2	2.0345	1.017251	1.97004	0.985021
4	3.88166	0.970414	3.93158	0.982896
8	7.57723	0.947154	7.58218	0.947772
16	13.9481	0.871754	14.271	0.891940
32	24.696	0.771749	28.4306	0.888456
64	39.5752	0.618362	51.5151	0.804923
128	57.4662	0.448955	85.2369	0.665914
256	74.957	0.292801	127.787	0.499168
512	85.6681	0.167320	170.06	0.332149
1024	92.4889	0.090321	204.124	0.199340
	4096x4096		**8192x8192**	
1	1.0	1.000000	-	-
2	1.99502	0.997509	-	-
4	3.29702	0.824255	-	-
8	6.56886	0.821107	-	-
16	12.9829	0.811433	-	-
32	25.4664	0.795824	-	-
64	49.7125	0.776758	-	-
128	92.9156	0.725903	-	-
256	153.001	0.597659	-	-
512	246.141	0.480744	-	-
1024	311.184	0.303890	-	-

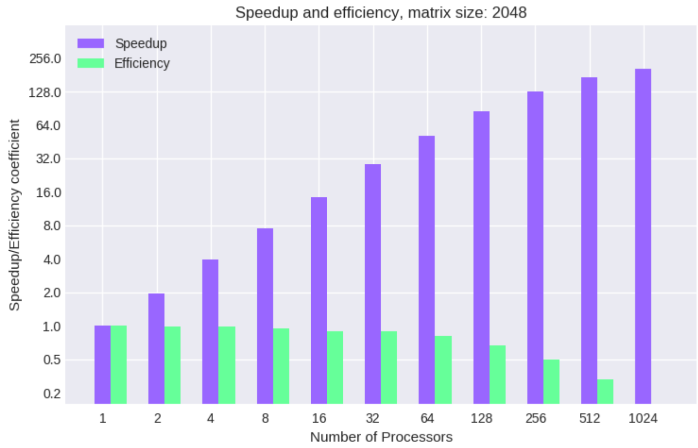
Графики

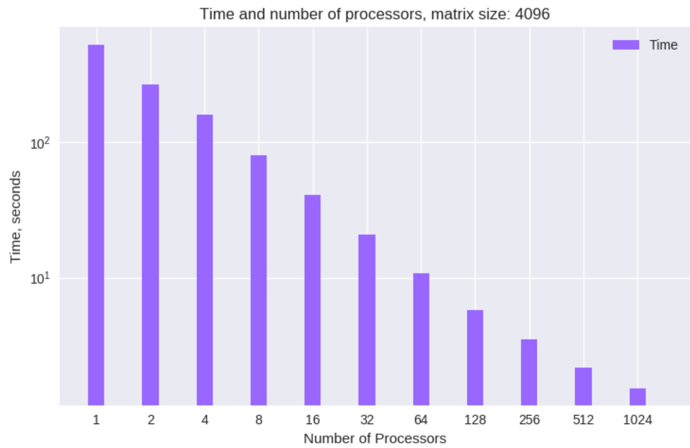
Далее приедены графики для матриц 1024x1024, 2048x2048, 4096x4096 и 8192x8192.

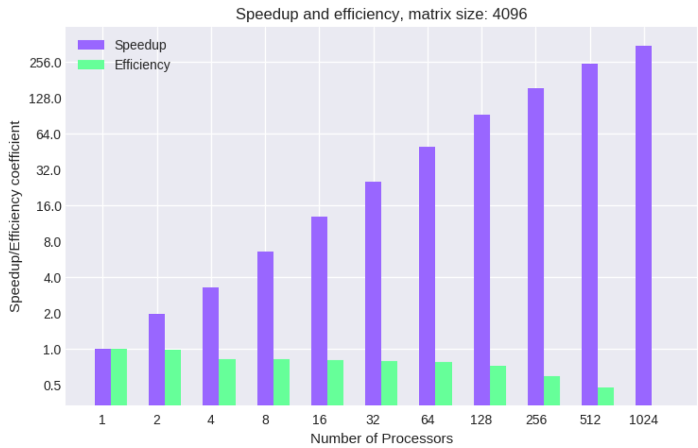


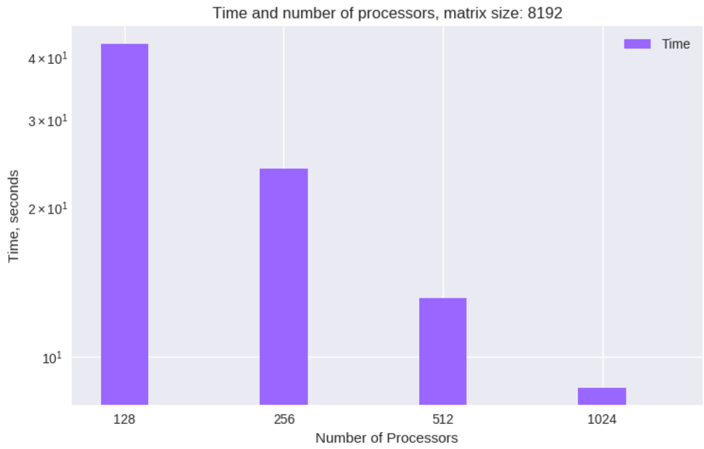


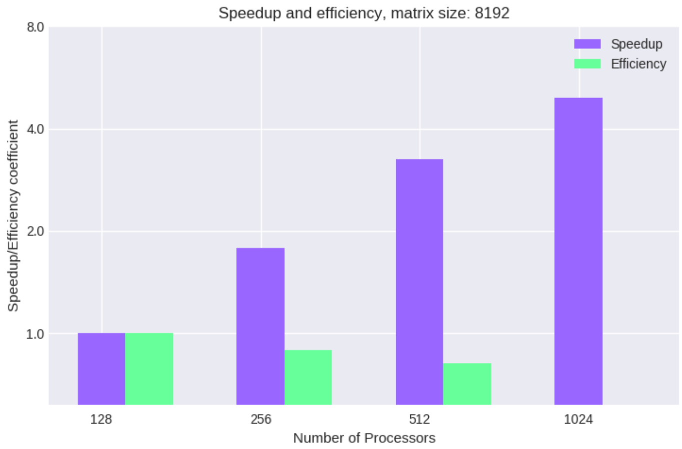












Основные выводы

С увеличением числа процессов время выполнения значительно уменьшается. Задача решения СЛАУ с помощью метода отражений эффективно распараллеливается, и можно получить значительное ускорение при достаточно высокой эффективности. При небольшом размере матрицы эффективность обратного хода метода Гаусса существенно снижается, по сравнению с большими матрицами. При увелечении числа процессов может наблюдаться суперлинейное ускорение в связи с тем, что части матрицы начинают полностью помещаться в кэш-линии L2 процессов.